Proceedings of the 2017 Winter Simulation Conference
W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, eds.

USING QUALITY OF SERVICE LANES TO CONTROL THE IMPACT OF RAID TRAFFIC WITHIN A BURST BUFFER

Elsa Gonsiorowski

Lawrence Livermore National Laboratory
Livermore, CA, USA

Christopher D. Carothers
Justin LaPre

Rensselaer Polytechnic Institute
Troy, NY, USA

Philip Heidelberger

IBM T.J. Watson Research Center
Yorktown Heights, NY, USA

Cyriel Minkenberg
German Rodriguez

Rockley Photonics, SWITZERLAND

ABSTRACT

The next generation of leadership supercomputer systems will require a medium-term layer of storage. The basis for this stratum of storage will be a Storage I/O Node (SION). For increased reliability, a redundancy algorithm will be implemented on top of groups of SIONs. In addition to the overheads of implementing a redundancy mechanism, a large cost of using a RAID strategy comes from the possibility of increased network congestion due to rebuild operations. To better understand the impact of RAID rebuild traffic, we have developed a simulation model of the SIONs. After validation, we use this model to investigate the impact of several configuration parameters, including redundancy mechanism and the physical arrangement of hardware. Additionally, our model analyzes the use of Quality of Service lanes to limit the impact of RAID traffic. We conclude with a series of recommendations for configuring a resilient and high performing I/O subsystem.

1 INTRODUCTION

The next generation of leadership supercomputer systems will require a medium-term layer of storage. This layer of storage should be reliable and responsive to applications running on the system and should persist for days or weeks. This storage will enable ensemble-jobs (multiple jobs for a single application/experiment), such as in-situ visualization or data analysis. Implemented with non-volatile storage, this layer exists between the compute system and parallel file system and is called a *burst buffer*.

The driver of this work is the U.S. Department of Energy (DOE) "DesignForward" program which supports research and development of technologies for Exascale system (Exascale Initiative 2015). The logical unit for this stratum of storage is a Storage I/O Node (SION). SIONs are two-way SMP nodes (meaning there exist two network endpoints per node) controlling NVRAM-based storage. For increased reliability, a redundancy algorithm will be implemented on top of groups of SIONs. In addition to the overheads of implementing a redundancy mechanism, a large cost of using a RAID strategy comes from the possibility of increased network congestion due to rebuild operations.

To better understand the impact of RAID rebuild traffic, we have developed a simulation model of the SIONs. We investigate two methods for redundancy in declustered network storage systems, Reed-Solomon (RS) error-correcting codes and Locally Repairable Codes (LRC), including the theoretical space and network utilization of each method. We implement a SION model within the IBM's OMNeT++ based simulator (Birke, Rodriguez, and Minkenberg 2012, Minkenberg and Rodriguez 2009, Varga and Hornig

978-1-5386-3428-8/17/$31.00 ©2017 IEEE

2008). We perform a brief analyses of our model to validate it and understand the optimal rates at which read, write, and rebuild operations could be performed within a single SION group.

This paper has three primary contributions. First, we develop an event-driven simulation of the SION model is developed using the IBM Venus network simulation framework and validated by comparing its performance to an analytical model. Second, through simulation experiments, this model exposes the impact of rebuild traffic on read and write traffic over a variety of parameters, including: RAID mechanism, SION group arrangement, quality-of-service (QoS) configuration, and number of SION RAID groups participating in a rebuild. Third, we perform an extensive set of simulation experiments to evaluate the impact of a reduced bisection bandwidth network configurations within the I/O subnetwork. A key finding is that these experiments show that a reduced bisection bandwidth network configuration, while a legitimate cost-saving measure, has unpredictable performance impacts with regards to network traffic latency and throughput when the QoS lanes are in use.

This paper is organized as follows. Section 2 begins with a brief review of related work. Section 3 and 4 give basic information on RAID and network simulation. Section 5 discusses the mathematical basis of our model followed by a description of the parameters studied in our experiments, Section 6. Finally, we present our experimental results in Section 7 and discuss our conclusions in Section 8.

2 RELATED WORK

As Exascale systems come closer to reality, there has increasing opportunities for research through simulations (Wang et al. 2014). These simulations can examine how the underlying network topology can impact the usage patters (Liu et al. 2012). Using these simulation models, there have been various studies and research activities on the software technologies which can use and manage the burst buffers (Herbein et al. 2016; Moody et al. 2010; Sato et al. 2014; Wang et al. 2014).

The HPC storage hierarchy will change drastically in the Exascale time frame (Lang et al. 2009; Zhao et al. 2013). These storage hierarchies must maintain high-throughput performance for both users of the parallel file systems and for applications within the supercomputer. This will be achieved through the use of an integrated burst buffer (Kimpe et al. 2012). Early burst buffer systems have found success for applications performing in-situ analysis and checkpoint/restart (Bhimji et al. 2016; Moody et al. 2010).

3 BURST BUFFER REDUNDANCY

The structure of our proposed NVRAM-storage system follows that of current parallel file systems for leadership class supercomputer systems (Schmuck and Haskin 2002; Corbett et al. 1993; May 2001) and will have a model similar to the one described in Rao, Hafner, and Golding (2011). An applications file which can be read from or written to in parallel by compute nodes (CNs) is divided into “blocks” as part of an overall declustered approach to network storage. A block is then divided into stripes. Next, to ensure that each write operation is optimal for the physical storage medium, each stripe is divided into N data strips. For today’s storage media, N range between 4 and 16 with a typical value being $N = 8$. Additionally, some number of parity strips are created. The type and number is determined by the reliability codes used and the level of fault tolerance required.

The SIONs are clustered into groups. When writing a new block to the NVRAM-storage, the approach will be to spread all the data and parity strips onto unique SIONs within a group. While there are I/O optimizations that can be done, our model makes the simplifying assumption that for any I/O operation, the whole block is either read or written. So, on a read of the data block, only the data strips for each stripe must be read, assuming no failures within that block. On a write or read-modify-write operation, the whole block is written (or re-written) and associated parities are recomputed for a stripes within the block.

Parallel storage systems were first implemented using arrays of RAID storage such as earlier versions of IBM’s GPFS. The term *RAID* has always represented the technology of using multiple independent disks for data redundancy (Arpaci-Dusseau and Arpaci-Dusseau 2012).

The drawback to a traditional RAID level approach is that the data rebuild rate is limited by the spare disk being used in the rebuild process. To break this bottleneck, multiple implementations of a declustered approach to networked storage now exist. These solutions uniformly partition and distribute data and parity strips across a declustered RAID group. Thus, for a rebuild operation a spare storage device is not used but instead all remaining storage devices within the RAID group participate in parallel in the rebuild process. This declustered, network storage approach that is the basis for our networked, NVRAM-storage system.

The mathematical basis for data reliability lies in the area of error-correcting codes within information theory. Due to its optimally least storage overhead, Reed-Solomon (RS) codes are often the go-to mechanisms for redundancy. RS codes are widely implemented and are built into the standards for data storage, data transmission, and bar codes (Wikipedia 2014). In contrast, there has been recent research in developing a network-efficient redundancy mechanism. The current solution is called Locally Repairable Codes (LRC). These codes have seen quick adoption by large corporations, such as Facebook (Khan et al. 2012).

4 NETWORK SIMULATION

The efficiency of file I/O operations can have a large impact on the perceived performance of a supercomputer. Typically, large amounts of data must be transferred from disk to the local CNs. Careful consideration must be made to ensure that the CN communication network does not become congested with I/O data, especially when a supercomputer is executing multiple, simultaneous jobs.

The easiest way to incorporate a storage layer into a compute network is to confine it to a subnetwork within the larger system. Depending on the exact wiring, the I/O subnetwork can minimize the impact of I/O operations on the overall CN communication network (as discussed in Chen et al. (2016)). By separating the SIONs into their own subnetwork, we have the added flexibility in choice of CN network. That is: the I/O subnetwork topology can be different than that of the CN network topology.

To implement the I/O subnetwork we chose to use a reliable and robust network topology: the Fat Tree. While the Fat Tree is robust with respect to both failures and traffic patterns, it is not the cheapest candidate topology. By allowing the I/O subnetwork to be realized in a different topology than that of the CN network, we have minimized the impact of the higher cost topology. I/O subnetwork is smaller than the CN network thus the cost-impact of using a Fat Tree is likely justified due to its increased robustness.

We base our network hardware, topology, and configurations on existing IBM simulation models, which are representative of today's hardware. To create our I/O subnetwork, we use two levels of 24 switches each with 48 ports (for a total of 48 switches in the network). Overall, the I/O subtree is connected to the CN-side of the network via 576 incoming links on the second level switches. The two levels of switches are connected by 576 dedicated links. Below the middle-of-rack (MOR) switches, there are total of 576 network endpoints, connected to a total of 288 SIONs (two ports for each dual-endpoint SMP node). We divide the total 288 SIONs into 12 RAID groups of 24 SIONs each. This configuration provides a balance between the group size and the number of groups and allows us to contain the impact of a failure.

The I/O subnetwork implementation allows for the CN-to-CN network topology and its traffic to be excluded from the experiments for this task. Therefore, we are only concerned with CN-to-SION traffic, modeled as read and write steams entering the I/O sub-tree. The behavior of this traffic is largely independent of the compute network topology. This work was done in parallel with Chen et al. (2016), which discusses several network topologies for next generation supercomputers. The I/O subnetwork used here is capable of being integrated with those topologies.

5 MODEL CHARACTERISTICS AND MATHEMATICAL BASIS

The overhead of fault tolerance is most felt during I/O operations. To better understand this overhead we constructed an analytical model. This model represents an optimum configuration and can be used to determine the steady-state, peak performance of SIONs during typical operations. We start by understanding how bandwidth of one group of SIONs reacts to basic operations (reads, writes, and rebuilds). We then investigate two arrangements of SION groups within a network topology. The "local" and "distributed" allocations each have their own benefits and drawbacks.

We study how read, write, and rebuild operations are handled by a group of SIONs. We assume that each SION in the group is participating in matching operations. We also assume an ideal scenario: a single data or parity strip is either stored (write) or exits (read) within the SION performing the operation.

Due to the distributed nature of the system, a simple I/O request is fulfilled by a SION group, rather than a single SION. This means that when a SION receives one such request, it must perform in-group communications. The precise rate of these communications depends on the RAID method being implemented. For this analysis we use RS(8,3) and LRC(8,5,4).

We analyze these operations in terms of uni-directional bandwidth, either B_{in} or B_{out}, of a single endpoint. We assume that the bandwidths are symmetric, that is that $B_{\text{in}} = B_{\text{out}}$. In order for a SION to fulfill a read, write, or rebuild request it must interact with other SIONs in its group. Due to the symmetric nature of the SION-group communications, incoming and outgoing bandwidths used for group communications must be held at same rate. Thus, these rates will affect the maximum rate at which the SION group can communicate with CNs in the system.

Table 1: Maximum read, write, and rebuild rates to a single SION in terms of the SION's bandwidth, B.

	$r_{\max}$	Overhead Factor	$w_{\max}$	Strips Needed	Rebuild Rate
Scheme		x	$B/(2+x)$	y	B/y
$RS(8,3)$	$(8/15)B_{\text{out}}$	0.25	$B/(2.25)$	8	$B/8$
$LRC(8,5,4)$	$(8/15)B_{\text{out}}$	0.5	$B/(2.5)$	4	$B/4$

5.1 Read Operations

A read request is a query for entire stripes of data and assumes there are no failures. A SION group responding to a set of read operations (one operation per SION in the group) sees the following two network communications.

First, each SION in the group must request 7 strips from across the group to reconstitute a single stripe of data. This is due to our distributed storage architecture where both the RS and LRC methods use 8 data strips per stripe (again, we are assuming that one strip is already present on the given SION). Here there is no difference in possible read bandwidths for RS and LRC. These in-group communications are symmetric on all SIONs; meaning as each individual SION receives 7 strips, it also distributes 7 strips to the group.

Second, each SION fulfills its read request with rate r by sending complete stripes (8 strips) of data.

Overall, this operation has the outgoing bandwidth, B_{out}, as the limiting factor. This is due to the fact that it must accommodate both CN and SION-group communications. We use this limitation to calculate the maximum achievable read rate. Given SION bandwidth B_{out} and the need to communicate $7/8$ strips with other SIONs in the group, our maximum read bandwidth is limited by our SION's outgoing bandwidth according to the relation: $B_{\text{out}} = r_{\max} + (7/8)r_{\max}$. This is equivalent to $(8/15)B_{\text{out}} = r_{\max}$. This result indicates that a SION can stream data to the CNs at maximum of 53% of its own outgoing bandwidth. This restriction is the result of declustered storage.

5.2 Write Operations

Write operations to a group of SIONs have three distinct elements.

First write requests, originating from the CNs within the system, are received by each SION in the group. The rate of a write request is denoted as w.

Second, each SION locally performs parity calculations. These calculations turn each stripe of write data into 8 data strips with either 3 or 5 additional parity strips. Thus, the write request for size s results in the actual storage of $s(1+p/8)$ data within the SION group (see Section 6.1). One strip is stored locally.

Finally, $8+p-1$ strips are distributed to other SIONS in the same group. If the original write request rate is w, then the amount of data leaving the SION can be represented as $w(1+x)$, where x depends on the RAID method. We can calculate that $x=0.25$ for RS(8,3) and $x=0.5$ for LRC(8,5,4). Once again, the symmetric nature of SION-group communications means that each SION also receives $w(1+x)$.

Overall, this operation has the incoming bandwidth, B_{in}, as the limiting factor. It must accommodate both incoming SION-group communications as well as the incoming write request data. Using this limiting factor, we can calculate the maximum achievable write rate for both RS and LRC. An incoming write request of rate w incurs incoming SION-group communication traffic of rate $w(1+x)$. Thus, our maximum write bandwidth is limited by a SION's incoming bandwidth according to the relation: $B_{\text{in}} = w_{\text{max}} + w_{\text{max}}(1+x)$. This is equivalent to $B_{\text{in}}/(2+x) = w_{\text{max}}$. This relation indicates that the particular RAID method in place impacts the maximum write bandwidth a single SION can sustain. These results are summarized in Table 1.

5.3 Rebuild Operations

A simple form of rebuild occurs when one SION in a group fails. The remaining members of the group participate in rebuilding the lost data. Each remaining SION must receive y strips ($y=8$ for RS and $y=4$ for LRC) in order to rebuild one of the lost strips. Once again, the symmetric nature of the SION-group communications means that each SION experiences the same incoming and outgoing bandwidth load. Thus, given the SION bandwidth B (either B_{in} or B_{out}), we can rebuild at a rate of B/y (See Table 1).

5.4 Quality of Service Limitations

In order to guarantee a certain QoS for I/O operations on the SIONs, network traffic associated with rebuild operations can be given a lower priority. We thus allocate a fraction, f, of a SION's bandwidth to be used for rebuild when both rebuild and I/O traffic occupy the network. Thus the equation for maximum read rate is $r_{\text{max}} = (1-f)*(8/15)B$ and the equation for maximum write rate is $w_{\text{max}} = (1-f)*B/(2+x)$. The rebuild rate formula is simply $f*B/y$. Note that typically a read *or* write operation occurs for a given SION group. When only one of the read, write, or rebuild operations are occurring for a SION group, the bandwidth limiting factor will not be needed.

6 PARAMETERS FOR STUDY

Within the two-level Fat Tree subnetwork, there are several ways to configure the SIONs and the network. This section discusses the following experimentation parameters:

- RAID mechanism: Reed-Solomon or Locally Repairable Codes
- SION group arrangement: local or distributed allocation
- Quality of Service bandwidth allocation: 95, 90, 75, 50, or 25 configuration
- Bisection bandwidth of the subnetwork: full or half bisection
- Number of SION RAID groups rebuilding in the system: 1 or 4

This leads to a total of 80 combinations. By analyzing the performance in each configuration, we are able to understand the impact of each option and the efficacy of a particular setup.

6.1 RAID Algorithms

When implementing a declustered network storage system, data availability and reliability are a principal concern. Error-correcting codes are used to break-up and store a single piece of data with extra "parity" information. Each chunk of data and parity is stored on one of the independent RAID devices. For this work we discuss a system that can withstand three device failures.

There are two widely used mechanisms for redundancy in declustered network storage: Reed-Solomon (RS) error-correcting codes and Locally Repairable Codes (LRC). These two methods are mathematically optimal solutions, each an optimization for a different constraint. RS optimizes for the least storage overhead whereas LRC optimizes for information locality.

Reed-Solomon codes are a common form of erasure codes (Reed and Solomon 1960). For any number of original data strips, *k*, RS creates parity strips: one per degree of fault tolerance. For our system with 3 degrees of fault tolerance we use RS(8,3), called $8data\ +\ 3parity$ or $8+3P$ in shorthand. To show that RS is part of an optimal family of codes, we start with a basic definition from coding theory:

Definition 1 (Minimum Code Distance) The minimum distance d of a code with n strips, is equal to the minimum number of erasures of strips after which the original stripe cannot be retrieved.

For RS(8,3) we have $n = 8+3 = 11$, (i.e., the total number of strips [per stripe] stored in the system) with a distance of $d = 3$. In fact, RS codes are optimal in this regard. For each additional parity strip generated by RS, there is an additional level of fault tolerance. The family codes with this characteristic are called Maximum Distance Separable (MDS) codes (Wicker and Bhargava 1999; Dimakis et al. 2011). It is also easy to conclude that MDS codes have the lowest impact on overall storage capacity.

We now introduce a second definition from coding theory:

Definition 2 (Block Locality) A code has locality r, when each strip is a function of at most r other strips from the stripe.

For RS codes, it takes a minimum of d strips to recreate any single strip. While RS is flexible in which strips are needed (one can use any combination of the data and parity strips), it creates high network load for each rebuild. It is this fact that has lead researchers to explore less network intensive codes.

The notion of *local* codes has been a recent area of research (Papailiopoulos and Dimakis 2012; Rawat et al. 2014; Huang, Chen, and Li 2007). The term "local" means only that the original set of data strips are split into two groups. Each of these groups receives some form of local parity.

For our work, we base LRC off Reed-Solomon codes with additional local parity strips, as presented in Sathiamoorthy et al. (2013). It is obvious that with additional local parities, LRC are not part of the MDS family. However, LRC are theoretically optimal in terms of information locality and are efficient with respect to network bandwidth as well as disk I/O for rebuilds.

To construct an LRC equivalent of RS(8,3) we start with the same 8 data strips. The data strips are divided into two local groups, each of which has a local parity. We add the additional constraint of an implied parity, which is not stored within the system. The implied parity acts as the local parity for a group of RS-generated parity strips. These local parities also satisfy an *alignment equation*: $L_1 + L_2 = L_3$ (Sathiamoorthy et al. 2013). It can be shown that the loss of any single strip requires only 4 other strips to rebuild it. In our configuration of 8 data strips and 3 degrees of fault tolerance, LRC requires 5 stored parity strips and has locality $r = 4$. This expressed as LRC(8,5,4).

Table 2 compares RS and LRC for an 8 data strip system which can withstand 3 losses.

Table 2: Space and network usage analysis of a system with 3 degrees of fault tolerance.

	Storage		Network Transmissions		
Scheme	Efficiency	Overhead	Stripe Read	Stripe Write	Strip Rebuild
$RS(8,3)$	27.27%	37.5%	8	11	8
$LRC(8,5,4)$	38.46%	62.5%	8	13	4

There are two ways to measure how a redundancy method affects space usage within a system. The first is a storage-centric measurement called *space efficiency* (Chen et al. 1994; Sathiamoorthy et al. 2013; Xin et al. 2003). This is percentage of total storage that will be devoted to redundancy. The second metric is data-centric and is called *storage overhead* ((Chen et al. 1994); Sathiamoorthy et al. 2013). This is the percent of additional space needed to store a given amount of data.

We calculate each metric with following formulas, where d is the number of data strips stored and p is the number of parity strips stored. Efficiency is equal to $p/(d+p)$ whereas overhead is equal to p/d. The results of these calculation for both RS and LRC can be seen in Table 2.

The recovery times of a system is largely based on the speed of the network connections between SIONs in the same group. For the rebuild of any stripe, we must collect some number of the remaining data and/or parity strips. For our 8 data strip system with 3 degrees of fault tolerance, RS requires twice as many strips per strip rebuild than LRC (8 strips versus 4). This means that in the scenario of transferring data through the network so that a single SION may execute a rebuild, we would expect that recovery time for RS would be twice as long as that for LRC.

It is important to note that the number of network requests during rebuild is not the only measurement of network efficiency. We must also consider how the network is used during read and write requests. For example, we expect that a single SION will execute the particular RAID technique to break a stripe of data into a number of data and parity strips. Since each method of RAID creates a different number of parity strips, there will be a differing amount of network traffic to disperse the strips to other SIONs in the group (for storage). Understanding traffic flow during read/write operations will contribute to understanding the advantages of each RAID method.

For the proposed two-level Fat Tree I/O network, there are 576 endpoints connecting to a total of 288 SIONs (dual-endpoint nodes). To decrease the overhead of performing a rebuild, the SIONs are divided into several RAID groups. An appropriate division of these 288 SIONs is 12 RAID groups consisting of 24 SIONs (or 48 endpoints). These SIONs can be physically connected to the network via the MOR switches in two ways: a local allocation and a distributed allocation.

In the local allocation, one RAID group of SIONs is contained within two MOR switches. Each SION in the group has one endpoint connection to each of the two switches, thus there are 24 SIONs (or 48 endpoints) per group. Overall, each SION group connects to all 24 downward-facing ports on the two local switches. As such, all intra-group communication is contained within a switch, for improved network efficiency. This configuration trades keeping intra-group traffic within a switch against low resiliency in the case of a MOR switch failure. This is an important configuration due to fact that all operations (rebuild, read, and write) require some intra-group communication. This traffic can be contained within the MOR switches for improved efficiency and no traffic will traverse to the second level switches. Overall, it is most efficient to locally contain all intra-group SION traffic. However, a result of the local allocation is that a single MOR switch failure results in a loss of 50% of the bandwidth from a single SION group.

In the distributed allocation, a SION group is distributed across all MOR switches. Every group has two ports on each MOR switch; inversely, one switch connects to two nodes from every group. For intra-group communications, the messages must travel outside of the MOR switches thus impacting other, global traffic. This configuration trades high resiliency against a MOR switch failure for global network traffic. In the event of such a failure, each SION group loses 4% of its bandwidth. A distributed allocation protects against multiple switch failures. It takes at least 4 MOR switch failures before a portion of data would become unavailable. The resiliency of the distributed allocation is in stark contrast to the local allocation. With the distributed allocation, the intra-group SION traffic must traverse to second-level switch. Overall, this communication pattern is much less efficient.

The two SION allocations within the network are each ideal for a particular usage scenario. Through experimentation, we obtain a detailed understanding of how the different configurations impact the overall responsiveness of the storage layer to CN requests. At a high level, we can assess some of the strengths and weaknesses of each allocation through one simple scenario: the failure of a middle of rack switch.

For the local allocation policy, a MOR switch failure is quite hazardous for a single SION group. The affected SION group would lose 50% of its bandwidth as one endpoint on each node is connect to the failed switch. In the absolute worst case, a second MOR switch failure would disconnect the entire SION group (all of the data it contained) from the network. Depending on the expected rate of MOR switch failure, this allocation could prove to be quite risky.

In contrast, the distributed allocation policy is much better suited to withstanding a MOR switch failure. In this configuration, each SION group loses 4% of its bandwidth. This configuration can also withstand many more MOR switch failures, up to three without any data loss or unavailability. In the worst case, the loss of a fourth MOR switch would mean the loss of four SIONs from every group, which is more than the three-degrees-of-fault-tolerance this RAID method can withstand.

6.2 Quality of Service

To limit the impact that SION rebuild traffic has within the network, we experiment with traffic quotas. These quotas determine the QoS for various types of traffic. For these experiments, we allocate a certain fraction of the bandwidth for rebuild traffic. By limiting how rebuild traffic interferes with read or write traffic on congested links, we can ensure the SION rebuild operations have minimal impact to the CNs.

We test five configurations for the QoS mechanism: 95, 90, 75, 50, and 25. These configurations correspond to the percentage of the bandwidth allocated to the read or write operations. The remaining bandwidth is allocated to the rebuild operation. We test a range of allocations in an attempt to balance the impact of rebuild traffic with the need to service read and write requests. Unless there is a high probability of critical failure (where a subsequent failure would cause data loss), read or write traffic should be allocated a majority of the bandwidth.

6.3 Bisection Bandwidth

With the Fat Tree topology there is an option available for increased cost-saving: reducing the bisection bandwidth by one half. By decreasing the number of links connecting the MOR and second level switches, we can decrease the overall cost of the I/O subnetwork. A half bisection topology is one where half of the second level switches have been removed. Consequently, half of the links connecting the MOR and second level switches are also removed. For the two-level Fat Tree topology described in Section 4, this means that there are only 12 second level switches, for a total of 36 switches (compared to 48 switches in the full bisection network). Overall, the half bisection network represents a 25% cost savings. Reducing bisection bandwidth is guaranteed to affect the overall network performance and utilization. By experimenting with reduced bisection bandwidth scenarios, we will understand the performance impact of this cost-saving measure. For our experiments we test both a full bisection Fat Tree as well as a Fat Tree with 50% of the regular bisection bandwidth.

6.4 Number of Rebuilding Groups

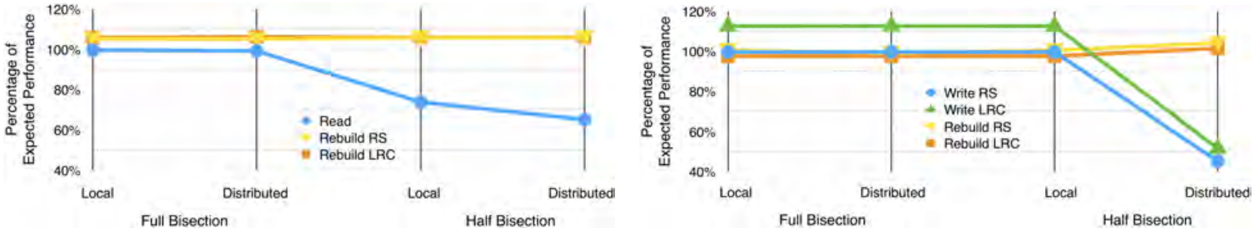
The impact of rebuild traffic on read or write traffic depends on the number of SION RAID groups participating in rebuild operations. Thus, this parameter ranges from 0 (when there are no SION groups performing a rebuild) to 12 (when all SION groups are performing a rebuild). For these experiments, we test scenarios with 1 or 4 SION groups participating in a rebuild. These rebuilds are the results of "soft-failures," where the SIONs have detected some local data corruption. All SIONs in the rebuilding groups are active and able to participate in the rebuild, read, and write operations.

7 EXPERIMENTAL RESULTS

A simulation model for the SION storage layer discussed here has been implemented on IBM's OMNeT++ based simulator (Birke, Rodriguez, and Minkenberg 2012, Minkenberg and Rodriguez 2009, Varga and Hornig 2008). This C++ based simulator performs high-fidelity network simulation, used for understanding the impact of network topology and routing algorithms in HPC systems. It has been used to study how different CN-network topologies impact network traffic in future Exascale systems (Chen et al. 2016). We used the IBM DmodK routing algorithm for all simulations.

For our experiments, we consider a "best case" example of failure, where no entire SION node has failed and all 24 SIONs in each RAID group are able to contribute to both the read/write and rebuild operations. Within this scenario, we tested the parameters defined in Section 6. These experiments measured the impact of each parameter on overall performance of the read and write operations in the presences of SION rebuild traffic. To test all combinations of the parameter values required a campaign of 80 individual simulations. We ran 3 campaigns to understand the impact to read, write (with RS), and write (with LRC) operations. This totaled 240 simulations.

In an effort to reduce the number of variables the analysis presented here, we compressed the QoS and number of rebuild groups dimensions. Through simulation, we found that these dimensions are highly predictable.



(a) Read Experiments

(b) Write Experiments

Figure 1: Parallel coordinates plot of read and rebuild-during-read and (b) write and rebuild-during-write performance. The performance is graphed over two parameters: SION allocation and bisection bandwidth. The performance is averaged over all other parameters and compared to the baseline peak performance.

Figures 1a and 1b present our simulation results as parallel coordinates graphs. Here, we present two dimensions for analysis: bisection bandwidth and SION allocation. Each vertical axis represents one set of experiments: (i) Full bisection, local SION allocation; (ii) Full bisection, distributed SION allocation; (iii) Half bisection, local SION allocation; and (iv) Half bisection, distributed SION allocation. On these axes, we plot the performance of read, write, and rebuild traffic (for both RS and LRC). In Figure 1a, there is only one plot for read performance, since it does not vary between RS and LRC (see Table 2).

One expects that experiments with full-bisection bandwidth of the I/O subnetwork will match closely with the analytical model prediction. Thus, we first analyze the experiments with a full-bisection subnetwork to demonstrate the accuracy of the analytical models (the two, left axes in Figures 1a and 1b).

For most of the full bisection bandwidth experiments, the performance results are within a narrow margin around the expected values predicted by the analytical model. For read operations, the experimental performance is over 98% of the expected performance. Rebuild operations during read perform slightly better than expected (approximately 105% of expected). Similarly, the performance of the write operation with RS are at least 99% of the expected performance. Surprisingly, the write operation with LRC performance is actually slightly better than expected. As shown by the green-triangle line in Figure 1b, these results are approximately 115% of the expected performance.

Using these graphs, we evaluate the performance of experiments where the I/O subnetwork is configured with 50% bisection bandwidth (the two, right axes in Figures 1a and 1b). In these half-bisection bandwidth scenarios, we would expect that the overly simplified analytical models do not accurately align with observed behavior. The fact that our simulation results for full bisection experiments closely match with the expected performance predicted by the analytical models, asserts that we can use the same simulation model for half bisection experiments.

For the half bisection bandwidth experiments, a much different performance picture emerges. In Figure 1a, we observe that performance under the both the local and distributed SION allocations is heavily impacted and is less than 80% of the expected performance. We attribute the approximately 10% performance difference between local placement and distributed placement to the loss bisection bandwidth for the distributed SION placement.

The write performance in the half bisection experiments show a drastic difference between local and distributed SION allocations (Figure 1b). Similar to the read rate data, here we observe that write operations (both under RS and LRC) perform in the local allocation close to expectations. In contrast, write operations in the distribute SION placement performs much lower than expected. Here, the performance is approximately 50% of what is expected. We attribute this sharp decline in write rate to the loss of bisection bandwidth in the next level of switches which the distributed SION placement traffic must use. In this same set of experiments, we see a corresponding increase in the performance of the rebuild operations. We attribute this phenomenon to the fact that the write operations are consuming less bandwidth so much so that it does not meet its QoS share. The additional available bandwidth is given to the rebuild traffic.

8 SUMMARY AND CONCLUSIONS

Through an analysis of 240 simulation experiments, we have determined the impact of a specific SION configuration has on a checkpoint/restart workload of an application running within the supercomputer.

For full bisection bandwidth scenarios, we demonstrate a minimal difference between the overall performance of a local SION allocation versus a distributed one. This, coupled with the ability to withstand multiple MOR switch failures, leads us to recommend the distributed SION allocation. Given the distributed SION allocation recommendation, and the poor performance of read and write operations within a half-bisection, distributed configuration, using a Fat Tree with full bisection bandwidth is highly recommended. The final variable, the RAID mechanism, yields a weaker recommendation. Essentially, it comes down to a cost/benefit analysis between storage space and network usage. For future Exascale systems, where the likelihood of network contention is high, and has the potential to dramatically impact to the end-user, we recommend LRC RAID. While LRC has a higher storage overhead, the huge decrease in network usage during rebuild scenarios is attractive.

Future work involves a continuing exploration of the range of parameters in this space. In particular, it should analyze the performance of simultaneous write and rebuild operations.

ACKNOWLEDGMENT

This work was done while Elsa Gonsiorowski was at Rensselaer Polytechnic Institute and while Cyriel Minkenberg, German Rodriguez, and Bogdan Prisacari were at IBM Zurich Research Laboratory. This work was supported and partially funded by Lawrence Berkeley National Laboratory (LBNL), on behalf of the US Department of Energy, under LBNL subcontract number 7078416. Lawrence Livermore National Laboratory is operated by Lawrence Livermore National Security, LLC, for the U.S. Department of Energy, National Nuclear Security Administration under Contract DE-AC52-07NA27344.

REFERENCES

Arpaci-Dusseau, R. H., and A. C. Arpaci-Dusseau. 2012. *Operating Systems: Three Easy Pieces*.

Bhimji, W., D. Bard, M. Romanus, D. Paul, A. Ovsyannikov, B. Friesen, M. Bryson, J. Correa, G. K. Lockwood, V. Tsulaia et al. 2016. “Accelerating Science with the NERSC Burst Buffer Early User Program”. Proceedings of Cray Users Group [Online]. Available: https://cug.org/proceedings/cug2016proceedings/includes/files/pap162.pdf.

Birke, R., G. Rodriguez, and C. Minkenberg. 2012. “Towards Massively Parallel Simulations of Massively Parallel High-performance Computing Systems”. In *Proceedings of the Fifth International ICST Conference on Simulation Tools and Techniques*, 291–298. Desenzano del Garda, Italy.

Chen, D., P. Heidelberger, C. Stunkel, Y. Sugawara, C. Minkenberg, B. Prisacari, and G. Rodriguez. 2016. “An Evaluation of Network Architectures for Next Generation Supercomputers”. In *Proceedings of the Seventh International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*, 11–21. Salt Lake City, UT, USA.

Chen, P. M., E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. 1994. “RAID: High-Performance, Reliable Secondary Storage”. *ACM Computing Surveys* 26:145–185.

Corbett, P. F., D. G. Feitelson, J.-P. Prost, and S. J. Baylor. 1993. “Parallel Access to Files in the Vesta File System”. In *Proceedings of the 1993 ACM/IEEE Conference on Supercomputing*, 472–481.

Dimakis, A., K. Ramchandran, Y. Wu, and C. Suh. 2011. “A Survey on Network Codes for Distributed Storage”. *Proceedings of the IEEE* 99 (3): 476–489.

Exascale Initiative 2015. “Design Forward”. (2015) [Online] Available: http://www.exascaleinitiative.org/design-forward Accessed on: 11 May 2017.

Herbein, S., D. H. Ahn, D. Lipari, T. R. Scogland, M. Stearman, M. Grondona, J. Garlick, B. Springmeyer, and M. Taufer. 2016. “Scalable I/O-Aware Job Scheduling for Burst Buffer Enabled HPC Clusters”. In *Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing*, 69–80. Kyoto, Japan.

Huang, C., M. Chen, and J. Li. 2007. “Pyramid Codes: Flexible Schemes to Trade Space for Access Efficiency in Reliable Data Storage Systems”. In *Proceedings of the Sixth IEEE International Symposium on Network Computing and Applications*, 79–86.

Khan, O., R. C. Burns, J. S. Plank, W. Pierce, and C. Huang. 2012. “Rethinking Erasure Codes for Cloud File Systems: Minimizing I/O for Recovery and Degraded Reads”. In *FAST*, 20.

Kimpe, D., K. Mohror, A. Moody, B. Van Essen, M. Gokhale, R. Ross, and B. R. de Supinski. 2012. “Integrated In-System Storage Architecture for High Performance Computing”. In *Proceedings of the Second International Workshop on Runtime and Operating Systems for Supercomputers*, 4:1–4:6. Venice, Italy.

Lang, S., P. Carns, R. Latham, R. Ross, K. Harms, and W. Allcock. 2009. “I/O Performance Challenges at Leadership Scale”. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, 40:1–40:12. Portland, OR, USA.

Liu, N., J. Cope, P. Carns, C. Carothers, R. Ross, G. Grider, A. Crume, and C. Maltzahn. 2012. “On the Role of Burst Buffers in Leadership-Class Storage Systems”. In *Proceedings of the 28th Symposium on Mass Storage Systems and Technologies*, 1–11.

May, J. M. 2001. *Parallel I/O for high performance computing*. Morgan Kaufmann.

Minkenberg, C., and G. Rodriguez. 2009. “Trace-driven Co-simulation of High-performance Computing Systems Using OMNeT++”. In *Proceedings of the Second International Conference on Simulation Tools and Techniques*, 65:1–65:8. Rome, Italy.

Moody, A., G. Bronevetsky, K. Mohror, and B. R. d. Supinski. 2010. “Design, Modeling, and Evaluation of a Scalable Multi-level Checkpointing System”. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–11. New Orleans, LA, USA.

Papailiopoulos, D., and A. Dimakis. 2012. “Locally Repairable Codes”. In *Proceedings of the 2012 IEEE International Symposium on Information Theory*, 2771–2775.

Rao, K. K., J. Hafner, and R. Golding. 2011. “Reliability for Networked Storage Nodes”. *IEEE Transactions on Dependable and Secure Computing* 8 (3): 404–418.

Rawat, A., O. Koyluoglu, N. Silberstein, and S. Vishwanath. 2014. “Optimal Locally Repairable and Secure Codes for Distributed Storage Systems”. *IEEE Transactions on Information Theory* 60 (1): 212–236.

Reed, I., and G. Solomon. 1960. “Polynomial Codes Over Certain Finite Fields”. *Journal of the Society for Industrial and Applied Mathematics* 8 (2): 300–304.

Sathiamoorthy, M., M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur. 2013. “XORing Elephants: Novel Erasure Codes for Big Data”. *Proceedings of the VLDB Endowment* 6 (5): 325–336.

Sato, K., K. Mohror, A. Moody, T. Gamblin, B. R. d. Supinski, N. Maruyama, and S. Matsuoka. 2014. “A User-Level InfiniBand-Based File System and Checkpoint Strategy for Burst Buffers”. In *Proceedings of the 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 21–30.

Schmuck, F. B., and R. L. Haskin. 2002. “GPFS: A Shared-Disk File System for Large Computing Clusters.”. In *FAST*, Volume 2, 19.

Varga, A., and R. Hornig. 2008. “An Overview of the OMNeT++ Simulation Environment”. In *Proceedings of the First International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, 60:1–60:10.

Wang, T., S. Oral, Y. Wang, B. Settlemyer, S. Atchley, and W. Yu. 2014. “BurstMem: A High-Performance Burst Buffer System for Scientific Applications”. In *Proceedings of the 2014 IEEE International Conference on Big Data*, 71–79.

Wicker, S. B., and V. K. Bhargava. 1999. *Reed-Solomon codes and their applications*. John Wiley & Sons.

Wikipedia 2014. “Reed-Solomon error correction — Wikipedia, The Free Encyclopedia”. [Online; accessed 11-September-2014].

Xin, Q., E. Miller, T. Schwarz, D. D. E. Long, S. Brandt, and W. Litwin. 2003. “Reliability mechanisms for very large storage systems”. In *PRoceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies*, 146–156.

Zhao, D., D. Zhang, K. Wang, and I. Raicu. 2013. “Exploring Reliability of Exascale Systems Through Simulations”. In *Proceedings of the High Performance Computing Symposium*, 1:1–1:9. San Diego, CA, USA.

AUTHOR BIOGRAPHIES

ELSA GONSIOROWSKI is an application I/O specialist and systems software developer at Lawrence Livermore National Laboratory. She received a Ph.D. in Computer Science from Rensselaer Polytechnic Institute, Troy, NY. Her research interests include software for application checkpointing and parallel discrete-event simulation. Her e-mail address is gonsie@llnl.gov.

CHRISTOPHER D. CAROTHERS is a professor in the Computer Science Department at Rensselaer Polytechnic Institute. He received the Ph.D., M.S., and B.S. from Georgia Institute of Technology in 1997, 1996, and 1991, respectively. His research interests are focused on massively parallel computing which involve the creation of high-fidelity models of extreme-scale networks and computer systems. His e-mail address is chrisc@cs.rpi.edu.

JUSTIN LAPRE is a lecturer at Rensselaer Polytechnic Institute, where he received his Ph.D. in Computer Science. His research interests include parallel discrete event simulation, compilers, and static analysis. His e-mail address is laprej@cs.rpi.edu.

PHILIP HEIDELBERGER received his B.A. in Mathematics from Oberlin College in 1974 and his Ph.D. in Operations Research from Stanford University in 1978. He was a Research Staff Member at the IBM T.J. Watson Research Center in Yorktown Heights, NY from 1978 to 2016, and he is currently a consultant at IBM Research. His research focused on highly efficient simulation algorithms, and more recently on supercomputer design as a member of IBM Research’s Blue Gene supercomputer hardware team. He has co-authored over 130 papers, seven of which have won outstanding paper awards including the 2006 ACM/IEEE Gordon Bell Prize. He is a co-inventor on over 70 U.S. Patents and a Fellow of the IEEE and ACM. His e-mail address is philiph@us.ibm.com.

CYRIEL MINKENBERG is a System Architect at Rockly Photonics. He received his doctorate Electrical Engineering in from Eindhoven University of Technology in Eindhoven, Netherlands His e-mail address is cyriel.minkenberg@rockleyphotonics.com.

GERMAN RODRIGUES is a System Architect at Rockly Photonics. He received his Ph.D. in Computer Architecture from Universitat Politecnica de Catalunya in Barcelona, Spain. His e-mail address is german.rodriguez@rockleyphotonics.com.